Survey of Network on Chip (NoC) Architectures & Contributions

Article *in* Journal of Engineering, Computing and Architecture · January 2009

CITATIONS
214

READS
4,034

2 authors:

Ankur Agarwal
Florida Atlantic University
79 PUBLICATIONS **678** CITATIONS
SEE PROFILE

Ravi Shankar
Florida Atlantic University
72 PUBLICATIONS **592** CITATIONS
SEE PROFILE

Journal of
Engineering, Computing and Architecture ISSN 1934-7197
Volume 3, Issue 1, 2009

Survey of Network on Chip (NoC) Architectures & Contributions

Ankur Agarwal, Florida Atlantic University, Boca Raton, FL, ankur@cse.fau.edu
Cyril Iskander, Hi-Tek Multisystems, Canada, cyril_iskander@hotmail.com
Ravi Shankar, Florida Atlantic University, Boca Raton, FL, ravi@cse.fau.edu

Abstract

Multiprocessor architectures and platforms have been introduced to extend the applicability of Moore's law. They depend on concurrency and synchronization in both software and hardware to enhance the design productivity and system performance. These platforms will also have to incorporate highly scalable, reusable, predictable, cost- and energy-efficient architectures. With the rapidly approaching billion transistors era, some of the main problems in deep sub-micron technologies which are characterized by gate lengths in the range of 60-90 nm, will arise from non-scalable wire delays, errors in signal integrity and un-synchronized communications. These problems may be overcome by the use of Network on Chip (NOC) architecture. In this paper, we have summarized over sixty research papers and contributions in NOC area.

Introduction

On a billion transistors chip, it may not be possible to send a global signal across the chip within real-time bounds [1]. If the SoC (System-on-Chip) is synchronized by a global clock signal, the circuit will be more prone to EMI (electromagnetic interference) [2]. The traditional system designs are usually based on critical paths and clock trees. These critical paths and clock trees contribute to an increased amount of power consumption. Therefore, SoCs are not power efficient. Besides, it is difficult to manage these clock trees due to clock skew problems [3].

As compared to synchronous designs, asynchronous designs are modular and do not suffer from issues such as clock skew, higher power consumption and EMI. However, designing asynchronous systems is a more complex task as compared to designing a synchronous system [4]. Designing a glitch free circuit and managing clock arrival time are complicated in the case of an asynchronous system. There is not much support from the EDA (Electronic Design Automation) industry for asynchronous systems. Thus, researchers have combined the ideas of synchronous and asynchronous designs. One such strategy is GALS (globally asynchronous and locally synchronous) solution. GALS divides a system into smaller, *locally decoupled synchronous regions* and then composes a few of them to yield a localized subsystem. These synchronous regions and subsystems would be easier to integrate into a global solution and verify. There will be an *asynchronous* way in which all the local synchronous regions will communicate at the system level. Therefore, these different synchronous regions need not have to be synchronized to a single global clock. This approach will reduce the requirement for chip-wide clock trees; the designers could focus on local synchronous regions only, which would be far less complex than the complete system. Since one has the flexibility to reduce the clock speed of a given synchronous region (or node) independent of other such regions, the amount of power consumption in a system can be managed better and reduced. One GALS solution is NOC (Network-on-Chip) [1]. NOC can improve design productivity by supporting modularity and reuse of complex cores. Thus, it enables a higher level of abstraction in the architectural modeling of future systems.

Related Research

In this section, we provide a detailed analysis of various contributions to the NOC domain.

Topology

From the communication perspective, there have been various topologies for NOC architecture. These include mesh, torus, ring, butterfly, octagon and irregular interconnection networks [5], [6]. Various researchers have

exploited these different NOC topologies for their NOC implementations. Kim *et al*. have used a star-based NOC that communicated using the principle of CDMA (Code Division Multiple Access) [7]; Adriahantenaina *et al*. proposed a tree-based implementation of NOC [8], where each node in the tree behaves as a router in NOC; Pande *et al*. compared various network topologies for interconnection networks in terms of latency, throughput, and energy dissipation [6]. Several researchers have suggested that a 2-D mesh architecture for NOC will be more efficient in terms of latency, power consumption and ease of implementation, as compared to other topologies. The Octagon NOC demonstrated in [9] is an example of a novel regular NOC topology.

Router Architecture

NOC architectures are based on packet-switched networks. This has led to new and efficient principles for design of routers for NOC [10]. Assume that a router for the mesh topology has four inputs and four outputs from/to other routers, and another input and output from/to the Network Interface (NI). Routers can implement various functionalities - from simple switching to intelligent routing. Since embedded systems are constrained in area and power consumption, but still need high data rates, routers must be designed with hardware usage in mind. For circuit-switched networks, routers may be designed with no queuing (buffering). For packet-switched networks, some amount of buffering is needed, to support bursty data transfers. Such data originate in multimedia applications such as video streaming. Buffers can be provided at the input, at the output, or at both input and output [11].

Various designs and implementations of router architectures based on different routing strategies have been proposed in the literature. Wolkotte *et al*. proposed a circuit switched router architecture for NOC [12], while Dally and Towles proposed a packet switched router architecture [13]. Albenes and Frederico provided a wormhole-based packet forwarding design for a NOC switch [14].

Routing Protocol

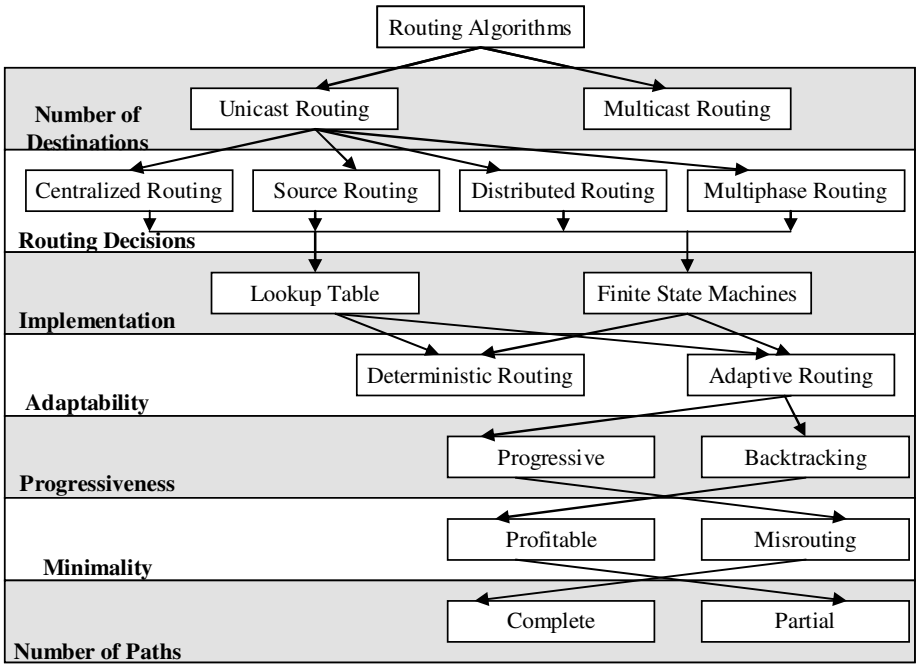

Figure 1. Routing algorithms.

Routing algorithms can be classified in various ways, as shown in Figure 1. In unicast routing, the packets have a single destination, while in the case of multicast routing, the packets have multiple destinations. For on-chip communication, unicast routing strategies seem to be a practical approach due to the presence of point-to-point

communication links among various components inside a chip. Based on the routing decision, unicast routing can be further classified into four classes: centralized routing, source routing, distributed routing and multiphase routing.

In centralized routing, a centralized controller controls the data flow in a system. In case of source routing, the routing decisions are taken at the point of data generation, while in distributed routing, the routing decisions are determined as the packets/flits flow through the network. The hybrid of the two schemes, source and destination routing, is called multiphase routing.

Routing algorithms can also be defined based on their implementation: lookup table and Finite State Machine (FSM). Lookup table routing algorithms are more popular in implementation. They are implemented in software, where a lookup table is stored in every node. We can change the routing algorithm by replacing the entries of the lookup table. FSM based routing algorithms may be implemented either in software or in hardware.

These routing algorithms may further be classified based on their adaptability. Deterministic routing always follows a deterministic path on the network. Examples of such routing algorithms are XY routing, North first, South first, East first, and West first. Adaptive routing algorithms need more information about the network to avoid congested paths in the network. These routing algorithms are obviously more complex to implement, thus, are more expensive in area, cost and power consumption. Therefore, we must consider a right QoS (Quality-of-Service) metric before employing these algorithms.

Routing algorithms can be fault-tolerant algorithms such as backtracking. In case of progressive algorithms, a channel is reserved before a flit is forwarded. Some routing algorithms send packets/flits only in the direction that is nearer to the destination. These routing algorithms are referred as profitable algorithms. A misrouting algorithm may forward a packet/flit away from the destination as well. Based on the number of available routing paths, routing algorithms can be finally classified as complete and partial routing algorithms.

Various routing algorithms have been proposed for the NOC. Most researchers suggested static routing algorithms and performed communication analysis based on the static behavior of NOC processes, thus, determining the static routing for NOC. Siebenborn *et al*. and Hu *et al*. used a CDG (Communication Dependency Graph) to analyze inter-process communications [15] [16].

Most NOC implementations used either XY routing or street sign routing algorithms. In [17], a comparison of deterministic (dimension-order) and adaptive routing algorithms for mesh, torus, and cube networks was presented. Mello *et al*. researched the performance of minimal routing protocol in NOC [18]. They concluded that the minimal routing provided better results than adaptive routing for on-chip-communications, as the adaptive routing concentrates on the traffic in the center of the NOC.

Switching Techniques

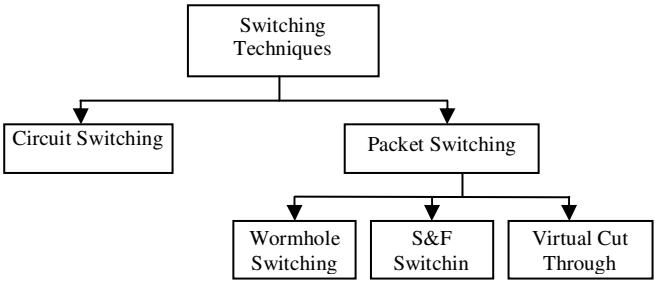

Figure 2. Switching techniques.

Switching techniques can be classified based on network characteristics. Circuit switched networks reserve a physical path before transmitting the data packets, while packet switched networks transmit the packets without reserving the entire path. Packet switched networks can further be classified as Wormhole, Store and Forward (S&F), and Virtual Cut Through Switching (VCT) networks (see Figure 2). In Wormhole switching networks, only the header flit experiences latency. Other flits belonging to the same packet simply follow the path taken by the header

flit. If the header flit is blocked then the entire packet is blocked. It does not require any buffering of the packet. Therefore, the size of the chip drastically reduces. However, the major drawback of this switching technique is a higher latency. Thus, it is not a suitable switching technique for real-time data transfers. Al-Tawil *et al*. provided a well-structured survey of Wormhole Routing techniques and its comparison with other switching techniques [19].

S&F switching forwards a packet only when there is enough space available in the receiving buffer to hold the entire packet. Thus, there is no need for dividing a packet into flits. This reduces the overhead, as it does not require circuits such as a flit builder, a flit decoder, a flit stripper and a flit sequencer. Nevertheless, such a switching technique requires a large amount of buffer space at each node. Thus, it may not be a feasible solution for embedded applications. The CLICHÉ implementation of a NOC is an example of store-and-forward switching [2]. Millberg *et al*. employed this switching technique in their Nostrum NOC implementation [20]. In VCT switching, a packet is forwarded to the next router as soon as there is enough space to hold the packet. However, unlike S&F, the VCT algorithm divides a packet into flits, which may be further divided into phits. Therefore, it has the same buffer requirement as S&F. None of the NOC implementations has adopted this switching technique in their implementation.

Ad-hoc switching techniques can be also developed by combining different switching techniques. For instance, VCs can be used for each class of traffic, while each channel is operated as per the principles of circuit switching. The Ethereal [10], [21] and Mongo NOC implementations use such a combination of techniques [22], [23].

Flow Control

Flow control determines how network resources, such as channel bandwidth, buffer capacity, and control state, are allocated to a packet traversing the network. The flow control may be buffered or bufferless (see Figure 3).

The Bufferless Flow Control has more latency and less throughput than thr Buffered Flow Control. The Buffered Flow Control can be further categorized into Credit Based Flow Control, ACK/NACK Flow Control, STALL/GO Flow Control, T-Error Flow Control, and Handshaking Signal based Flow Control.

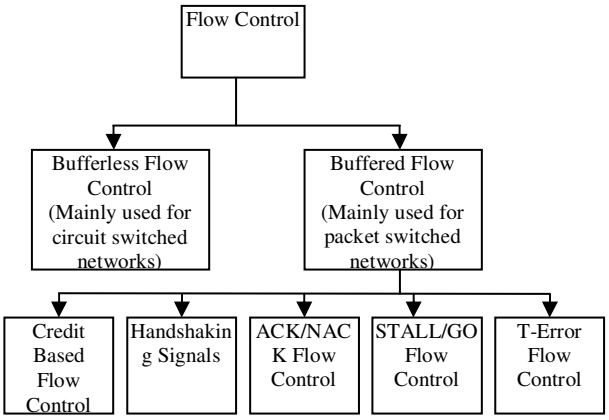

Figure 3. Flow control techniques.

In Credit Based Flow Control, an upstream node keeps count of data transfers, and thus the available free slots are termed as credits. Once the transmitted data packet is either consumed or further transmitted, a credit is sent back. Bolotin *et al*. used Credit Based Flow Control in QNOC [24], [25].

In Handshaking Signal Based Flow Control, a VALID signal is sent whenever a sender transmits any flit. The receiver acknowledges by asserting a VALID signal after consuming the data flit. Zeferino *et al*. used handshaking signals in their SoCIN NOC implementation [26].

In the ACK/NACK protocol a copy of a data flit is kept in a buffer until an ACK signal is received. On assertion of ACK, the flit is deleted from the buffer; instead if a NACK signal is asserted then the flit is scheduled for

retransmission. Bertozzi, Benini, and Micheli used this flow control technique in their XPIPES implementation [27], [28], [29].

In the STALL/GO scheme, two wires are used for flow control between each pair of sender (producer) and receiver (consumer). When there is an empty buffer space, a GO signal is activated. Upon the unavailability of buffer space, a STALL signal is activated. None of the present NOC implementations have employed this flow control scheme.

The T-Error Flow Control scheme is very complex as compared to other flow control mechanisms. It aims at enhancing the performance at the cost of reliability. Real time systems operating in a noisy environment must avoid the use of this flow control mechanism. None of the present NOC implementations has employed this flow control scheme.

Virtual Channel

The design of a virtual channel (VC) is another important aspect of NOC. A virtual channel splits a single channel into two channels, virtually providing two paths for the packets to be routed. There can be two to eight virtual channels. The use of VCs reduces the network latency at the expense of area, power consumption, and production cost of the NOC implementation. However, there are various other added advantages offered by VCs.

Network deadlock/livelock: Since VCs provide more than one output path per channel there is a lesser probability that the network will suffer from a deadlock; the network livelock probability is eliminated (these deadlock and livelock are different from the architectural deadlock and livelock, which are due to violations in inter-process communications).

Performance improvement: A packet/flit waiting to be transmitted from an input/output port of a router/switch will have to wait if that port of the router/switch is busy. However, VCs can provide another virtual path for the packets to be transmitted through that route, thereby improving the performance of the network.
Supporting guaranteed traffic: A VC may be reserved for the higher priority traffic, thereby guaranteeing the low latency for high priority data flits [30], [24].

Reduced wire cost: In today's technology the wire costs are almost the same as that of the gates. It is likely that in the future the cost of wires will dominate. Thus, it is important to use the wires effectively, to reduce the cost of a system. A virtual channel provides an alternative path for data traffic, thus it uses the wires more effectively for data transmission. Therefore, we can reduce the wire width on a system (number of parallel wires for data transmission). For example, we may choose to use 32 bits instead of 64 bits. Therefore, the cost of the wires and the system will be reduced.

Bjerregaard and Sparso have proposed the design and implementation of a virtual channel router using asynchronous circuit techniques [23], [24].

Buffer Implementation

A higher buffer capacity and a larger number of virtual channels in the buffer will reduce network contention, thereby reducing latency. However, buffers are area hungry, and their use needs to be carefully studied and optimized. Zimmer *et al*. and Bolotin *et al*. proposed a simple implementation of a buffer architecture for NOC [31], [32]. Zimmer *et al*. implemented buffers using 0.18 µm technology to estimate the cost and area of buffers needed for NOC. The Proteo implementation of a buffer architecture has been described in [33]. Gupta *et al*. studied the trade-off between buffer size and channel bandwidth to secure constant latency. They concluded that increasing the channel bandwidth is preferable to reducing the latency in NOC.

Error Correction and Decoding

The need for implementation of fault tolerant, error detection, and error correction techniques is not certain for on-chip implementations. Frederico, Santo, and Susin proposed a fault tolerant routing protocol for NOC [14]. Bolotin *et al*. in their implementation of QNOC [24], [25] argued that the communication strategies for on chip network may be considered reliable, while [34] and [35] proposed a fault tolerant routing algorithm and fault tolerant flow control

techniques for NOC architecture respectively. Zimmer, Jantasch, and Bertozzi, Binini, and Micheli proposed error detection and correction schemes for data on NOC links [36], [37].

Transmission Lines and Links

Another important component of a NOC is the design of interconnects. Barger *et al*. proposed a transmission line based design of interconnects for NOC [38]. Morgenshtein *et al*. compared serial and parallel links for interconnect implementations [39].

Network Interface

The network interface (NI) is responsible for packetization and depacketization of data traffic, in addition to conventional interfacing. This functionality may be implemented either with hardware or with software. Bhojwani and Mahapatra [40] compared software and hardware implementations of NI. They showed that the software implementation of NI takes about 47 cycles to complete packetization/depacketization, while the hardware version takes only 2 cycles. Substantial research has been conducted to propose the right data formats needed for various layers in the protocol stack. Ethereal and XPIPES NOCs use the OCP protocol, while SPIN and Proteo NOC have integrated the Virtual Component Interface (VCI) protocol in their implementations.

QoS

New algorithms have been proposed in this domain to reduce power consumption and area requirements while securing cost optimization [41]. One of the main concerns in NOC is to be able to reduce the latency of operation. Therefore, there are various levels of latency metrics that may be offered. Router architectures for supporting GT (Guaranteed Bandwidth) and BE (Best Effort) services have been proposed [10].

Arbitration Techniques

A NOC, which is capable of supporting different classes of service levels such as best effort and guaranteed traffic, needs to support an arbitration mechanism. This arbitration mechanism schedules a flit for transmission on the output path. There are various arbitration mechanisms such as RR (Round Robin), FCFS (First Come First Serve), PB (Priority Based), and PRBB (Priority Based Round Robin). Usually, RR and FCFS are used for best effort data flits and PB or PBRR is used for guaranteed traffic. SPIN [8], [41] and RASoC implemented RR arbitration, while QNOC [22], [23], XPIPES [27], [28], [29], and the Philips NOC [10], [21] have employed PBRR arbitration.

Architectural Issues

A NOC system may be categorized based on the customization and parameterization capabilities embedded in its architecture. A NOC architecture may further be defined as a homogenous architecture or a heterogeneous architecture. A heterogeneous architecture will have a fixed topology and cannot be customized as per an application requirement. Therefore, the design time with such an architecture will be less. However, a homogenous architecture may be customized each time as per the application requirement and may be more efficient in terms of area, power and latency of operation. Most of the NOC implementations support a homogenous architecture while the XPIPES supports a heterogeneous architecture [27], [28].

Most researchers have focused on the communication architecture of NOC. Binini and Micheli mapped the OSI layered architecture onto a NOC for on-chip communication. Agarwal and Shankar focused on exploiting the computing capability and provided a layered architecture for system design [42]. Their layered architecture consists of mapping different domains such as applications, algorithms, RTOS and protocol on a NOC environment. Agarwal and Shankar further applied a concurrency-modeling methodology to NOC. Their proposed architecture consists of concurrency-compliant reusable and parametrizable components. Most of the NOC architectures have yet to be implemented for commercial purposes. The Arteris NOC is an example of a commercially available NOC [3]. Arteris has developed a NOC complier for targeting applications onto its NOC chip. STMicroelectronics has shown interest in NOC implementations as well. Philips has implemented the Ethereal NOC (discussed later in this paper).

Once the design of the basic NOC architecture became established, new techniques evolved to address advanced issues such as dynamic load balancing, shortest/fastest data path, and energy-efficient NOC architecture design.

Mapping of Applications and Applications for NOC

Few researchers provide graph-based application decomposition and mapping strategies for NOC [16], [43]. Madasen *et al*. have proposed a RTOS (Real Time Operating System) architecture for NOC [44]. They provided a RTOS-based application scheduling technique on a NOC. Others have provided graph-based algorithms for application mapping onto a NOC platform [46], [47], [48]. NOC will not be a useful concept until we are able to demonstrate that an application performs better on an NOC as compared to a bus based communication infrastructure. Several researchers have studied this issue. For example, a LDPC (Low Density Parity Check) decoder was implemented on a NOC platform [48]. More than 60% of the hardware area of the LDPC is occupied by memory. The NOC architecture was thus shown to be a suitable platform for this application. Jiang, Wolf and Chanradhar implemented a video application on a NOC [49]. There is a need for a NOC emulation platform in order to study the impact of various NOC topologies on applications. Genko *et al*. provide a FPGA-based emulation platform for NOC [50].

Specific NOC Implementation

In this section we review and discuss some specific NOC implementations found in the open literature.

XPIPES NOC

The researchers of XPIPES [27], [28], [29] have generated a framework, XPIPES Compiler, which automatically instantiates customized NOC macros (switches, network interfaces and links) from the developed parametrizable building blocks implemented in SystemC. A static routing protocol called “street sign” routing along with wormhole switching are employed for on-chip communication. XPIPES uses pipelined links, similar to a shift register in operation, achieved by partitioning the wires into segments for the actual flit transfer. Each output module is deeply pipelined. The CRC (Cyclic Redundancy Check) decoders for error detection work in parallel with the switch operation. The first pipeline stage checks the headers of incoming packets on the different input ports to determine the correctness of the packet paths, the second pipeline stage resolves contention based on a round-robin policy. Arbitration is carried out when the tail flit of the preceding packet is received. A negative acknowledgement (NACK) for flits of non-selected packets is generated. The following arbitration stage keeps the status of the virtual channel registers and determines whether flits can be stored into the registers or not. The fifth stage is the actual buffering stage, and the ACK/NACK response at this stage indicates whether a flit has been successfully stored or not. The following stage takes care of forward flow control. Finally, a last arbitration stage multiplexes the virtual channels on the physical output link on a flit-by-flit basis.

The XPIPES network interface uses the standardized OCP interface to network cores. Static routing information is accessed by the header builder. It is then passed to the flit builder circuit in the form of a number of hops (NumSB) and an actual direction bit (LutWord) along with the datastream, if BusyBuilder is not asserted high. This flit is then passed to the NOC through the output buffer stage. The response path includes receiving information through Synchro, which reads only useful information and passes it to the core through Receive Response. XPIPES implements an error control logic based on the retransmission of data packets upon the negative acknowledgement.

QNOC

QNOC [22], [23] aims at providing different levels of quality of service for the end users. The architecture of QNOC is based on a regular mesh topology. It makes use of wormhole packet routing. Packets are forwarded using the static X-Y coordinate-based routing. It does not provide any support for error correction logic and all links and data transfers are assumed to be reliable. Packets are forwarded based on the number of credits remaining in the next router. QNOC has identified four different service levels (SL) based on the on-chip communication requirements. These SLs include Signaling, Real-Time, Read/Write (RD/WR) and Block Transfer, Signaling being the top priority and Block transfer being the least in the order as listed. The Priority Based Round-Robin scheduling criterion is

employed for transmission of flits. The cost functions for the QNOC implementation were calculated based on the estimation of area occupied by its components. Were also provided other performance parameters such as clock rate, end-to-end delay (latency of packet), and power consumption under different traffic loads.

Ethereal NOC

The Ethereal NOC developed by Phillips aims at achieving composability and predictability in system design and eliminating uncertainties in interconnects, by providing guaranteed throughput and latency services [10], [21]. It provides run-time reconfiguration. The Phillips NOC has an instance of a 6-port router with an area of 0:175 mm^2 after layout, and a network interface with four IP ports having a synthesized area of 0:172 mm^2. All the queues are 32-bits wide and 8-words deep. Hardware FIFOs (First In First Out) are used for implementing queues. Both the router and the network interface are implemented in 0:13 µm technology, and run at 500 MHz. The NI is able to deliver the BW of 16 Gbits/sec to all the routers in the respective directions. It is a topology-independent NOC. The NOC mainly consists of two components: the NI and the router, with multiple links between them. The NI offers the standard interface, such as AXI or OCP to the IP modules connected to the NOC. The NOC provides BE and GT service levels. A time-division multiplexed circuit switching approach with contention-free routing has been employed for GT. It uses wormhole routing with input queuing to route the flits. The router exploits source routing. GT flits are always scheduled for being routed in the next clock cycle, whereas BE flits are scheduled as per a round-robin criterion. Credit-based end-to-end flow control has been implemented to make sure that no flit is transmitted unless there is enough space in the destination buffer to accommodate it. The Phillips NOC implements the NI in two parts: the NI kernel and the NI shell. The NI kernel communicates with the NI shell via ports.

SPIN NOC

The Scalable Programmable Integrated Network-on-chip (SPIN) is based on a fat-tree topology [8], [41]. It addresses design decisions such as the nature of links, the packet structure and the network protocol. It is based on two kinds of components: initiators and targets. The system can have different numbers of cores for each type. The initiator components are traffic generators, which send requests to the target components. The target component sends a response as soon as it receives a request. All the components in the system are designed to be VCI (Virtual Socket Interface) compliant. SPIN is a packet-switching on-chip micro-network, which uses wormhole switching, adaptive routing and credit-based flow control. It is based on a fat-tree topology, which is a tree structure with routers on the nodes and terminals on the leaves, except that every node has replicated fathers. In a full 4-ary fat-tree topology, there are as many fathers as children on all nodes (routers). Such a topology produces a non-blocking network with a performance that scales gracefully with the system size. Links are bi-directional and full-duplex, with two unidirectional channels. The channel's width is 36 bits wide, with 32 data bits and 4 tag bits used for packet framing, parity and error signaling. Additionally, there are two flow control signals used to regulate the traffic on the channel. In SPIN, packets are defined as sequences of 32 bits data words, with the header fitting in the first word. An 8-bit field in the header is used to identify the destination terminal, allowing the network to scale up to 256 terminals. The payload has an unlimited length as defined by two framing bits (Begin Packet / End of Packet). SPIN uses wormhole switching. The input buffers have a depth of 4 words, which results in cheaper routers. Routing in SPIN is adaptive and distributed. The basic building block of the SPIN network is the RSPIN router. It includes eight ports, each port with a pair of input and output channels compliant with the SPIN link. Internally, RSPIN includes a 4 words buffer at each input channel and two 18 words output buffers, shared by the output channels. They have a greater priority when competing with the input buffer to use an output channel, which allows reducing the contention, whilst minimizing the head-of-line blocking by freeing the queues in the input buffers. RSPIN contains a partial 10×10 crossbar, which implements only the connections allowed by the routing scheme: all the packets flowing down the tree can be forwarded to children and only such packets can use the output buffers when the required output channel is busy. Nevertheless, only the packets incoming from children can flow up the tree and be forwarded to the fathers.

Other NOCs

There have been a sizeable number of proposals/implementations of NOCs in the literature. Without being exhaustive, these include:

A. A mesh-based NOC using Chip-Level Integration of Communication Heterogeneous Elements (CLICHÉ) [2];
B. Proteo, a flexible-topology NOC [33];
C. A guaranteed-throughput switch for a circuit-switched NOC, which supports both unicast and multicast [51];
D. *HiNoc,* a NOC offering both a low-overhead transmission service and one with guaranteed QoS, with a two-level asynchronous/synchronous hierarchy [32];
E. A reconfigurable circuit-switched NOC [12];
F. NOCs for the Princeton Smart Camera SoC [49];
G. A CMOS implementation of a NOC interconnecting multiple processing units of different clock frequencies [52];
H. MANGO (Message-passing Asynchronous Network-on-Chip providing Guaranteed services through OCP interfaces) [22] [23];
I. A connectionless NOC implementing Diffserv to provide QoS [53] [54];
J. SoCBUS (Switched Network on Chip for Hard real time embedded systems) aims at providing the guaranteed throughput [55];
K. SoCIN, a parametric and scalable NoC [26] based on the RASoC router soft-core [56], and its evolution SoCINfp, based on the ParIS switch [57].

NOC Issues and Challenges

To enhance system productivity, it is very important that an architect be able to abstract, represent and address most of the design issues and concerns at a high level of abstraction. System-level design affords one the opportunity to review several different software-hardware architectures that meet the functional specifications equally well, to quickly trade-off among different QoS metrics such as latency, power, cost, size and ease of integration. Similarly, there are several issues related to NOC, such as the nature of the NOC link, link length, serial vs parallel links, bus vs packet-based switching, and leakage currents. In this section, we discuss these issues.

Serial vs Parallel Link

The transportation of data packets among various cores in a NOC can be performed by the use of either a serial or a parallel link. Parallel links make use of a buffer-based architecture and can be operated at a relatively lower clock rate in order to reduce power dissipation. However, these parallel links will incur high silicon cost due to inter-wire spacing, shielding and repeaters. This can be minimized up to a certain limit by employing multiple metal layers. On the other hand, serial links allow savings in wire area, reduction in signal interference and noise, and further eliminate the need for having buffers. However, serial links would need serializer and de-serializer circuits to convert the data into the right format to be transported over the link and back to the cores. Serial links offer the advantages of a simpler layout and simpler timing verification. Serial links sometimes suffer from ISI (Inter-symbol Interference) between successive signals while operating at high clock rates. Nevertheless, such drawbacks can be addressed by encoding and with asynchronous communication protocols.

Interconnect Optimization

Communication in a NOC is based on modules connected via a network of routers with links between the routers that comprise of long interconnects. Thus it is very important to optimize interconnects in order to achieve the required system performance. Timing optimization of global wires is typically performed by repeater insertion. Repeaters result in a significant increase in cost, area, and power consumption. Recent studies indicate that in the near future, inverters operating as repeaters [58] will use a large portion of chip resources. Thus, there is a need for optimizing power on the NOC. Techniques for reducing dynamic power consumption include approaches discussed in [59], [60], [61]. Encoding is another effective way of reducing dynamic power consumption [62]. In order to make NOC architectures more effective, innovative ways will have to be introduced to minimize the power consumed by the on-chip repeaters.

Leakage Power Consumption

The leakage current, which was negligible relative to the dynamic switching current at larger transistor sizes (of 1 micron or more), is expected to dominate the current drain at sub-100 nm technologies. In a NOC, the link utilization rates vary and in many cases are very low, reaching a few percentage points. Networks are designed to

operate at low link utilization in order to meet worst case scenario requirements, and thus having a higher link capacity helps reduce packet collisions. However, even when NOC links are idle they still will consume power in repeaters, due to the dominance of this leakage current at small feature sizes. Thus, new techniques will have to evolve which will help reduce the leakage power consumption to make the NOC architecture more effective.

Router Architecture

For embedded systems such as handheld devices, cost is a major driving force for the success of the product and therefore the underlying architecture as well. Along with being cost effective, handheld systems are required to be of small size and to consume significantly less power, relative to desktop systems. Under such considerations, there is a clear tradeoff in the design of a routing protocol. A complex routing protocol would further complicate the design of the router. This will consume more power and area without being cost effective. A simpler routing protocol will outperform in terms of cost and power consumption, but will be less effective in routing traffic across the system.

Quality of Service Challenges

QoS parameters for a NOC include latency, cost, power consumption, and silicon overhead in terms of the area that is required to support the packet switched network. It is expected that various applications such as real-time, multimedia and control, and computation-intensive algorithms such as video encoding and decoding algorithms, 3-D gaming, and speech recognition, will be supported on a NOC environment. Under such a scenario, NOC should be able to provide various levels of support for these applications.

The NOC infrastructure must be able to guarantee a timely exchange of data packets for a real-time application. However, given a certain network size, large latency fluctuations for packet delivery could be experienced because of network congestion. Such variability and non-determinacy are obviously not acceptable for real-time applications. One possible solution to this problem is over-dimensioning the network by adding some redundant paths (links), nodes and buffers. These paths can be utilized when the main network is congested. Another possible solution to this problem would be to reserve some paths for real-time applications in order to guarantee a timely delivery of data packets from one node to another. Both solutions are able to overcome the latency problem but increase the power consumption and the cost of NOC. A cost effective solution would be to provide priority levels to the data traffic. Real-time traffic can be guaranteed its on-time delivery to its destination by either reserving some paths for real-time data, or implementing priority-based scheduling criteria. In such systems, we would be able to route the data packet for a real-time application in time but a data packet belonging to a lower priority application may be starved, without appropriate scheduling algorithms.

The NOC infrastructure includes components responsible for packetization, transmission, and de-packetization of data. These components, respectively, are the NI, the VC router, and the links. These components are repeated for every grid element in NOC. So, if we consider a NOC with 3×3 mesh network, then it will have nine sets of components of NI, VC router and links. It can be clearly seen that these components will occupy a significant amount of silicon space on the chip and therefore the cost and the power consumption of the chip would increase. However, it must be noted that serial packet-based communication will still remain an optimum solution as compared to a bus-based system in terms of the power consumption and will reduce the cost of system design in the longer run due to the potential for reuse.

Consideration for System-Level Simulation Environments

A simulation environment must allow us to integrate hundreds of cores and model concurrency and synchronization issues. This requires the representation of various activities and algorithms with appropriate MoCs (Models of Computation). We define here the five main issues that should be met for a simulation environment to be suitable for system-level designs. The simulation model must allow a system architect to: (1) model the system functionality well in advance of building the actual computing system; (2) model concurrency issues among various components in the system model; (3) manipulate an abstract set of design elements simultaneously to generate different sets of QoS parameters and performance metrics, and to fine-tune the system model; (4) integrate different MoCs onto this

modeling environment; (5) access a well defined library of components defined in various MoCs so that the modeling time can be substantially reduced.

Various simulation environments have been used for modeling NOC architectures. NS-2 was developed for inter-computer networks, RSIM was developed for multiprocessor parallel computing, while other simulators were developed from scratch (e.g. NOCSim, Orion, PoPNet, and the re-targetable modeling and simulation platform using the Liberty Simulation Environment). Various researchers have used OPNET for simulation of on-chip traffic network.

Conclusion

The NOC concept elegantly separates the concerns of computing and communication, and is expected to be ideally suited to address this increased system complexity and declining system productivity. Researchers have well addressed NOC architectures and hardware-related issues. Still, an integrated approach for modeling, co-designing and co-developing HW-SW with a NOC architecture is missing. Application mapping strategies and feasible applications for NOC are other important aspects that need to be addressed in more detail. We need to research low cost, area and power efficient solutions of NOC for it to be applicable in the embedded systems industry.

References

[1] A. Jantsch and H. Tenhunen, *Network on Chips*, Kluwer Academic Publishers, Boston, 2003.

[2] S. Kumar, A. Jantsch, J-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, “A network on chip architecture and design methodology”, *IEEE Computer*, pp. 117-124, 2002.

[3] ARTERIS. 2005. *A comparison of network-on-chip and buses*. White paper. http://www.arteris.com/noc whitepaper.pdf.

[4] K. Emerson, “Asynchronous design - An interesting alternative”, *Proc. 10th International IEEE Conference on VLSI Design*, 1997, pp. 318-320.

[5] J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[6] P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, “Performance evaluation and design trade-offs for network-on-chip interconnect architectures”, *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025-1040, 2005.

[7] D. Kim, Manho Kim, and G.E. Sobelman, “CDMA-based NoC architecture”, *Proc. IEEE Conference on Circuits and Systems*, vol. 1, pp. 137-140, 2004.

[8] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C.A. Zeferino, “SPIN: a scalable, packet switched, on-chip micro-network, *Proc. IEEE Conference on Design, Automation and Test*, pp. 70-73, 2003.

[9] F. Karim A. Nguyen, and S. Dey, “An interconnect architecture for networking systems on chips”, IEEE Journal on Micro High Performance Interconnect, vol. 22, issue 5, pp. 36-45, Sept 2002.

[10] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, “Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip”, *IEE Proc. on Computers and Digital Techniques*, vol. 150, Issue 5, pp. 294-302, September 2003.

[11] A. Kumar, D. Manjunath, and J. Kuri, *Communication Networking: An Analytical Approach*, Morgan Kaufmann, 2004.

[12] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, “An energy-efficient reconfigurable circuit-switched network-on-chip”, *Proc. 19th IEEE International Conference on Parallel and Distributed Processing Symposium*, pp. 155-163, 2005.

[13] J. W. Dally and B. Towles, “Route packets, not wires: On-Chip interconnection networks”, Proc. *IEEE International Conference on Design and Automation*, pp. 684-689, June 2001.

[14] C. Albenes, Zeferino Frederico G. M. E. Santo, Altarniro Amadeu Susin, “ParIS: A parameterizable interconnect switch for Networks-on-Chips”, *Proc. ACM Conference*, pp. 204-209, 2004.

[15] A. Siebenborn, O. Bringmann, and W. Rosenstiel, “Communication analysis for network-on-chip design”, *Proc. IEEE International conference on Parallel Computing in Electrical Engineering*, pp. 315-320, 2004.

[16] J. Hu and R. Marculescu, “Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints”, *Proc. IEEE Conference Design Automation and Test in Europe*, vol. 1, pp. 234-239, 2004.

[17] C. Neeb, M. Thul, and N. Andwehn “Network on-chip-centric approach to interleaving in high throughput channel decoders”, *Proc. IEEE International Symposium on Circuits and Systems*, pp. 1766–1769, 2005.

[18] F. Moraes and N. Calazan, “An infrastructure for low area overhead packet-switching network on chip”, *Integration - The VLSI Journal,* vol. 38, Issue 1, pp. 69-93, October 2004.

[19] K. M. Al-Tawil, M. Abd-El-Barr, and F. Ashraf, “A survey and comparison of wormhole routing techniques in mesh networks”, *IEEE Network*, vol. 11, pp. 38–45, 1997.

[20] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch. “The Nostrum backbone - A communication protocol stack for networks on chip”, *Proc. IEEE International Conference on VLSI Design*, pp. 693, 2004.

[21] K. Goossens, J. Dielissen, and A. Rădulescu, “A Ethereal network on chip: Concepts, architectures, and implementations”, IEEE *Design & Test of Computers*, vol. 22, Issue 5, pp. 414-421, September 2005.

[22] T. Bjerregaard and J. Sparso, “Virtual channel designs for guaranteeing bandwidth in asynchronous Network-on-Chip”, *Proc. of IEEE Norchip Conference*, pp. 269 – 272, November 2004.

[23] T. Bjerregaard and J. Sparsø, “A router architecture for connection-oriented service guarantees in the MANGO clockless Network-on-Chip”, *Proc. Of IEEE on Design Automation and Test*, vol. 2, pp. 1226-1231, 2005.

[24] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, “QNoC: QoS architecture and design process for network on chip”, *Journal of Systems Architecture*, Volume 50, Issue 2-3 (Special Issue on Network on Chip), pp. 105-128, February 2004.

[25] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, “Cost considerations in Network on Chip”, *Integration: The VLSI Journal*, no. 38, 2004, pp. 19-42.

[26] C. A. Zeferino and A. A. Susin, “SoCIN: A parametric and scalable network-on-chip”, *Proc. 16th Symposium on Integrated Circuits and Systems Design*, pp. 169-175, 2003.

[27] D. Bertozzi and L. Benini, “Xpipes: A network-on-chip architecture for gigascale systems-on-chip”, *IEEE Circuits and Systems Magazine*, vol. 4, Issue 2, pp. 18-31, 2004.

[28] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, “NoC synthesis flow for customized domain specific multiprocessor systems-on-chip”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113-129, 2005.

[29] M. Dall'Ossa, G. Biccari, L. Giovannini, L. D. Bertozzi, and L. Benini, "XPIPES: A latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs", *Proc. 21st International IEEE Conference on Computer Design*, pp. 536-539, 2003.

[30] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework", *Proc. 11th International Symposium on Asynchronous Circuits and Systems* (ASYNC), pp. 54–63, 2005.

[31] E. Bolotin, A. Morgenshtein, I. Cidon, R. Ginosar, and A. Kolodny, "Automatic hardware-efficient SoC integration by QoS Network-on-Chip", *Proc. 11th International IEEE Conference on Electronics, Circuits and Systems*, pp. 479-482, 2004.

[32] H. Zimmer, S. Zink, T. Hollstein, and M. Glesner, "Buffer-architecture exploration for routers in a hierarchical network-on-chip", Proc. 19th IEEE International Symposium on Parallel and Distributed Processing, pp., 1-4, April 2005.

[33] I. Saastamoinen, M. Alho, and J. Nurmi, "Buffer implementation for Proteo network-on-chip", International IEEE Proceeding on Circuits and Systems, vol. 2, pp. 113-116, May 2003.

[34] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Fault tolerant algorithms for network-on-chip interconnect", Proc. IEEE Proceeding on Computer Society, pp. 46-51, February 2004.

[35] A. Pullini, Federico Angiolini, D. Bertozzi, and L. Benini, "Fault tolerance overhead in network-on-chip flow control schemes", *Proc. ACM Conference*, pp. 224-229, 2005.

[36] H. Zimmer and A. Jantsch, "A fault model notation and error-control scheme for switch-to-switch buses in a network-on-chip", *Proc. First International IEEE/ACM/IFIP Conference on Hardware/Software Codesign and System Synthesis*, pp. 188-193, 2003.

[37] D. Bertozzi, L. Benini, and G. De Micheli, "Error control schemes for on-chip communication links: the energy-reliability tradeoff", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818-831, 2005.

[38] A. Barger, D. Goren, and A. Kolodny, "Design and modeling of network on chip interconnects using transmission lines", *Proc. 11th IEEE International Conference on Electronics, Circuits and Systems*, pp. 403-406, 2004.

[39] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar, "Comparative analysis of serial vs parallel links in NoC", *Proc. IEEE International Conference on System-on-Chip*, pp. 185-188, November 2004.

[40] P. Bhojwani and R. Mahapatra, "Interfacing cores with on-chip packet-switched networks", *Proc. 16th International IEEE Conference on VLSI Design*, pp. 382–387, 2003.

[41] P. Bhojwani, R. Mahapatra, J. K. Eun, and T. Chen, "A heuristic for peak power constrained design of network-on-chip (NoC) based multimode systems", *Proc. IEEE International Conference on VLSI Design*, pp. 124-129, 2005.

[42] A. Agarwal and R. Shankar, "A layered architecture for NOC design methodology", *IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 659-666, 2005.

[43] U. Y. Ogras and R. Marculescu, "Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach", *Proc. IEEE Conference and Exhibition on Design, Automation and Test in Europe*, vol. 1, pp. 352-357, 2005.

[44] J. Madsen, S. Mahadevan, K. Virk, and M. Gonzalez, "Network-on-chip modeling for system-level multiprocessor simulation", *Proc. IEEE 14th Conference on Real-Time Systems*, pp. 265-274, 2003.

[45] L. Tang and S. Kumar, "Algorithms and tools for network on chip based system design", 16th IEEE Proc. on Integrated Circuits and Systems Design, pp. 163-168, Sept. 2003.

[46] K. Srinivasan, K. S. Chatha, and G. Konjevod, "Linear programming based techniques for synthesis of network-on-chip architectures", *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 422-429, 2004.

[47] R. Chae-Eun, J. Han-You, and Ha Soonhoi, "Many-to-many core-switch mapping in 2-D mesh NoC architectures", *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 438-443, Oct. 2004.

[48] T. Theocharides, G. Link, N. Vijaykrishnan, and M. J. Irwin, "Implementing LDPC decoding on network-on-chip", *Proc. 18th IEEE International Conference on VLSI Design*, pp. 134-137, 2005.

[49] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, and T. Lv. "A case study in networks-on-chip design for embedded video", IEEE *Proc. on Design Automation and Test in Europe Conference*, vol. 2, pp. 770-775, February 2004.

[50] N. Genko, D. Atienza, G. De Micheli, J. M. Mendias, R. Hermida, and F. Catthoor, "A complete network-on-chip emulation framework", *Proc. European IEEE Conference and Exhibition on Design, Automation and Test*, vol. 1, pp.246-251, 2005.

[51] J. Liu, L.-R. Zheng, and H. Tenhunen, "A guaranteed-throughput switch for network-on-chip", *Proc. the International Symposium on System-on-Chip*, pp. 31-34, 2003.

[52] S. J. Lee, K. Lee, S. J. Song, and H.-J. Yoo, "Packet-switched on-chip interconnection network for system-on-chip applications", *IEEE Trans. on Circuits and Systems-II*, vol. 52, no. 6, pp. 308-312, 2005.

[53] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Providing QoS to connection-less packet-switched NoC by implementing DiffServ functionalities", *Proc. 2004 International Symposium on System-on-Chip*, pp. 37-40, 2004.

[54] M. D. Harmanci, N. P. Escudero, Y. Leblebici, and P. Ienne, "Quantitative modeling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip", *Proc. 2005 International Symposium on Circuits and Systems*, vol. 2, pp. 1782-1785, 2005.

[55] D. Wiklund, L. Dake, "SoCBUS: Switched Network on Chip for Hard real Time Embedded Systems", IEEE International Proceedings of Parallel and Distributed Processing symposium, pp. 1-8, April 2003.

[56] C. A. Zeferino, M. E. Kreutz, and A. A. Susin, "RASoC: A router soft-core for networks-on-chip", Proc. Design Automation and Test in Europe Conference, vol. 3, pp. 198-203, 2004.

[57] C. A. Zeferino, M. E. Santo, and A. A. Susin, "ParIS: A parameterizable interconnect switch for networks-on-chip", Proc. 17th Symposium on Integrated Circuits and Systems Design (SBCCI'04), pp. 204-209, 2004.

[58] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar, "Low-leakage repeaters for NoC interconnects", Proc. IEEE International Symposium on Circuits and Systems, ISCAS 2005, vol. 1, pp. 600-603.

[59] N. Chabini and W. Wolf, “Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling”, IEEE Transactions on VLSI Systems, vol. 12, no. 6, pp. 573-589, 2004.

[60] L. Yan, L. Jiong, and N. K. Jha, “Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems”, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 24, no. 7, pp.1030-1041, 2005.

[61] L. Yan, J. Luo, and N. K. Jha, “Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems”, Proc. IEEE International Conference on Computer Aided Design, pp. 30-37, 2003.

[62] C. G. Lyuh and K. Taewhan, “Low power bus encoding with crosstalk delay elimination [SoC]”, 15th Annual Proc. IEEE International Conference on ASIC/SOC, pp. 389-393, September 2002.